WHAT IS CLAIMED IS:

1. A computer-implemented method for using a processor as a virtual device, said method comprising:

signaling, from a first processor, a second processor from a plurality processors in a computer system, wherein the plurality of processors share a common memory and wherein at least two of the processors are dislike;

storing data corresponding to the request in a local memory corresponding to the second processor; and

processing the data by the second processor using software code stored in the second processor's local memory.

2. The method as described in claim 1 further comprising:

writing, from a software program executing on the first processor, the data into an input buffer stored in the common memory, wherein the storing further includes sending the data from the input buffer into the second processor's local memory; and

loading the software code from the common memory to the second processor's local memory prior to the processing.

3. The method as described in claim 2 wherein the sending of the data into the second processor's local memory and the loading of the software code are both performed using DMA operations.

4. The method as described in claim 3 wherein the DMA operations are performed using one of a plurality of DMA controllers, one of which being assigned to the first processor and a second of which being assigned to the second processor.

5. The method as described in claim 1 further comprising:

generating result data, stored in the second processor's local memory, in response to the processing.

6. The method as described in claim 5 further comprising:

writing the result data to an output buffer stored in the common memory.

7. The method as described in claim 5 further comprising:

writing the result data to an input buffer that corresponds with another device.

8. The method as described in claim 7 further comprising:

signaling, from the second processor, a third processor from the plurality processors, wherein the third processor includes a local memory;

storing the result data in the third processor's local memory; and

processing the result data by the third processor using software code stored in the third processor's local memory.

9. The method as described in claim 5 further comprising:

writing the result data to a physical device.

10. The method as described in claim 1 wherein the software code stored in the second processor's local memory includes a plurality of code routines, the method further comprising:

identifying one of the code routines based upon the request; and

executing the identified code routine.

11. The method as described in claim 1 further comprising:

initializing a task queue for one or more device functions to be performed by one or more of the plurality of processors, wherein the signaling further includes:

writing the request to the task queue;

determining, by the second processor, that the request is in the task queue; and

reading the request from the task queue in response to the determination.

12. The method as described in claim 1 further comprising:

generating result data, stored in the second processor's local memory, in response to the processing.

13. The method as described in claim 1 further comprising:

prior to the signaling:

loading the data to a first location in the common memory;

loading the software code adapted to perform the processing to a second location in the common memory;

writing an instruction block to a third location in the common memory, wherein the instruction block includes the first and second locations;

wherein the signaling includes writing the third location to a mailbox corresponding to the second processor.

14. The method as described in claim 13 further comprising:

reading, by the second processor, the third location from the second processor's mailbox;

reading, by the second processor, the first and second locations from the instruction block stored at the third location; and

reading, by the second processor, the data from the first location in the common memory prior to storing the data in the second processor's local memory.

15. The method as described in claim 13 further comprising:

reading, by the second processor, the software code stored at the second location in the common memory; and storing the software code in the second processor's local memory.

16. The method as described in claim 15 wherein the reading and storing of the software code are performed in response to determining that the software code stored at the second location in the common memory is not already stored in the second processor's local memory.

17. An information handling system comprising:

a plurality of heterogeneous processors;

a common memory shared by the plurality of heterogeneous processors;

a first processor selected from the plurality of processors that sends a request to a second processor, the second processor also being selected from the plurality of processors;

a local memory corresponding to the second processor;

a DMA controller associated with the second processor, the DMA controller adapted to transfer data between the common memory and the second processor's local memory; and

a virtual device tool for operating the second processor as a virtual device, the virtual device tool including software effective to:

signal, from the first processor, the second processor;

store data corresponding to the request in the second processor's local; and

process the data by the second processor using software code stored in the second processor's local memory.

18. The information handling system as described in claim 17 further comprising software effective to:

write, from a software program executing on the first processor, the data into an input buffer stored in the common memory, wherein the storing further includes sending the data from the input buffer into the second processor's local memory; and

load the software code from the common memory to the second processor's local memory prior to the processing.

19. The information handling system as described in claim 18 wherein the sending of the data into the second processor's local memory and the loading of the software code are both performed using DMA operations with the DMA controller.

20. The information handling system as described in claim 17 further comprising software effective to:

generate result data, stored in the second processor's local memory, in response to processing the data using the software code.

21. The information handling system as described in claim 20 further comprising software effective to:

write the result data to an output buffer stored in the common memory.

22. The information handling system as described in claim 20 further comprising software effective to:

write the result data to an input buffer that corresponds with another device.

23. The information handling system as described in claim 22 further comprising software effective to:

signal, from the second processor, a third processor from the plurality processors, wherein the third processor includes a local memory;

store the result data in the third processor's local memory; and

process the result data by the third processor using software code stored in the third processor's local memory.

24. The information handling system as described in claim 20 further comprising software effective to:

write the result data to a physical device.

25. The information handling system as described in claim 17 wherein the software code stored in the second processor's local memory includes a plurality of code

routines, the information handling system further comprising software effective to:

identify one of the code routines based upon the request; and

execute the identified code routine.

26. The information handling system as described in claim 17 further comprising software effective to:

initialize a task queue for one or more device functions to be performed by one or more of the plurality of processors, wherein the signal of the second processor further includes software effective to:

write the request to the task queue;

determine, by the second processor, that the request is in the task queue; and

read the request from the task queue in response to the determination.

27. The information handling system as described in claim 17 further comprising software effective to:

generate result data, stored in the second processor's local memory, in response to the processing of the data using the software code.

28. The information handling system as described in claim 17 further comprising, prior to the signal of the second processor, software effective to:

loading the data to a first location in the common memory;

loading the software code adapted to perform the processing to a second location in the common memory;

writing an instruction block to a third location in the common memory, wherein the instruction block includes the first and second locations;

wherein the signal of the second processor includes software code effective to write the third location to a mailbox corresponding to the second processor.

29. The information handling system as described in claim 28 further comprising software to:

read, by the second processor, the third location from the second processor's mailbox;

read, by the second processor, the first and second locations from the instruction block stored at the third location; and

read, by the second processor, the data from the first location in the common memory prior to storing the data in the second processor's local memory.

30. The information handling system as described in claim 28 further comprising software effective to:

read, by the second processor, the software code stored at the second location in the common memory; and store the software code in the second processor's local memory.

31. The information handling system as described in claim 30 wherein the reading and storing of the software code are performed in response to determining that the software code stored at the second location in the common memory is not already stored in the second processor's local memory.

32. A computer program product stored on a computer operable media for using a processor as a virtual device, said computer program product comprising:

means for signaling, from a first processor, a second processor from a plurality processors in a computer system, wherein the plurality of processors share a common memory and wherein at least two of the processors are dislike;

means for storing data corresponding to the request in a local memory corresponding to the second processor; and

means for processing the data by the second processor using software code stored in the second processor's local memory.

33. The computer program product as described in claim 32 further comprising:

means for writing, from a software program executing on the first processor, the data into an input buffer stored in the common memory, wherein the means for storing further includes means for sending the data from the input buffer into the second processor's local memory; and

means for loading the software code from the common memory to the second processor's local memory prior to the processing.

34. The computer program product as described in claim 33 wherein the means for sending the data into the second processor's local memory and the means for loading the software code are both performed using DMA operations.

35. The computer program product as described in claim 34 wherein the DMA operations are performed using one of a plurality of DMA controllers, one of which being assigned to the first processor and a second of which being assigned to the second processor.

36. The computer program product as described in claim 32 further comprising:

means for generating result data, stored in the second processor's local memory, in response to the processing.

37. The computer program product as described in claim 36 further comprising:

means for writing the result data to an output buffer stored in the common memory.

38. The computer program product as described in claim 36 further comprising:

means for writing the result data to an input buffer that corresponds with another device.

39. The computer program product as described in claim 38 further comprising:

means for signaling, from the second processor, a third processor from the plurality processors, wherein the third processor includes a local memory;

means for storing the result data in the third processor's local memory; and

means for processing the result data by the third processor using software code stored in the third processor's local memory.

40. The computer program product as described in claim 36 further comprising:

means for writing the result data to a physical device.

41. The computer program product as described in claim 32 wherein the software code stored in the second processor's local memory includes a plurality of code routines, the computer program product further comprising:

means for identifying one of the code routines based upon the request; and

means for executing the identified code routine.

42. The computer program product as described in claim 32 further comprising:

means for initializing a task queue for one or more device functions to be performed by one or more of the plurality of processors, wherein the signaling further includes:

means for writing the request to the task queue;

means for determining, by the second processor, that the request is in the task queue; and

means for reading the request from the task queue in response to the determination.

43. The computer program product as described in claim 32 further comprising:

means for generating result data, stored in the second processor's local memory, in response to the processing.

44. The computer program product as described in claim 32 further comprising:

prior to the means for signaling:

means for loading the data to a first location in the common memory;

means for loading the software code adapted to perform the processing to a second location in the common memory; and

means for writing an instruction block to a third location in the common memory, wherein the instruction block includes the first and second locations;

wherein the means for signaling includes means for writing the third location to a mailbox corresponding to the second processor.

45. The computer program product as described in claim 44 further comprising:

means for reading, by the second processor, the third location from the second processor's mailbox;

means for reading, by the second processor, the first and second locations from the instruction block stored at the third location; and

means for reading, by the second processor, the data from the first location in the common memory prior to storing the data in the second processor's local memory.

46. The computer program product as described in claim 44 further comprising:

means for reading, by the second processor, the software code stored at the second location in the common memory; and

means for storing the software code in the second processor's local memory.

47. The computer program product as described in claim 46 wherein the means for reading and means for storing the software code are performed in response to determining that the software code stored at the second location in the common memory is not already stored in the second processor's local memory.